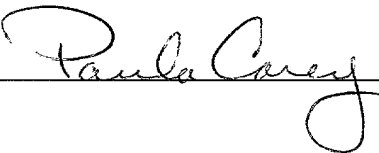


In Re Apln. of: Prell et al.  
Serial No.: To be assigned  
Filed: To be assigned  
For: METHOD FOR MANAGING THE SIMULTANEOUS UTILIZATION  
OF DIVERSE REAL-TIME COLLABORATIVE SOFTWARE  
APPLICATIONS

CERTIFICATE OF MAILING  
UNDER 37 CFR 1.10

THE UNDERSIGNED CERTIFIES THAT THIS PATENT APPLICATION IS BEING  
DEPOSITED AS EXPRESS MAIL, (LABEL NUMBER EL766614105US) ADDRESSED TO  
BOX PATENT APPLICATION, ASSISTANT COMMISSIONER FOR PATENTS,  
WASHINGTON, D.C. 20231, THIS 6<sup>th</sup> DAY OF April, 2001.

  
\_\_\_\_\_

METHOD FOR MANAGING THE SIMULTANEOUS UTILIZATION OF DIVERSE  
REAL-TIME COLLABORATIVE SOFTWARE APPLICATIONS

BACKGROUND OF THE INVENTION

5           This application claims the benefit of U.S. Provisional Application No. 60/196,021, filed April 7, 2000.

(a) Field of the Invention

          The present invention describes a method for managing the simultaneous utilization of diverse real-time  
10 collaborative software applications. This invention allows linking of multiple computer operators or users in a single collaborative session, involving multiple collaborative applications, and is transparent to the operator or user.

(b) Description of the Prior Art

15           The prior art methods used by computer professionals to provide a collaborative computer-based session between remote computer workstations have numerous shortcomings. One approach is to provide a mechanism for exchanging streaming data (video and voice) while concurrently exchanging  
20 application data. This is implemented by sharing one instance of an application on one of the participant's computers. For example, this approach was used by the creators of MS™ Net Meeting™. This approach operates by letting a computer operator, running a non-collaborative  
25 software application such as MS Word™, share the application with other people in the session. Users then take turns editing or controlling the single application. This approach does not work when the application in use in a session is a truly collaborative one, requiring multiple computer  
30 operators to exchange application data in real-time. An example of this type of application is any multi-player game.

          Another common approach is to provide a mechanism in an application supporting the exchange of limited data formats. For example, the application may support a real-time text  
35 message exchange (chat) and data exchange of some proprietary

data format. For example, the application may allow drawing of the same picture by different computer operators simultaneously. Unfortunately, this is all that application will support. It does not provide support for any third party applications employing alternate data formats.

#### SUMMARY OF THE INVENTION

The present invention relates to a real-time collaborative application which is a multi-computer operator application (computer program) that allows computer operators or users running this program on their computers to exchange application specific data in real-time.

The present invention allows one communication session to handle all communication links between diverse collaborative applications virtually transparently for the computer operator or user. The present invention achieves this without creating a bandwidth bottleneck at one central site.

A collaborative application session maintains some common data and exchanges some data between a particular group of computer operators participating in the session. Usually, collaborative applications are implemented using the client-server model, where a server is a separate computer program that hosts and/or implements one or more application sessions. Each computer operator uses a client program to connect to a particular application session on a server with a known network address and the server manages data sharing and exchange between computer operators participating in the session. Generally, collaborative applications use proprietary data formats and communications protocols. An example of a collaborative application is a multi-player computer game such as Quake III Arena™ or Unreal Tournament™, where computer operators play against each other, requiring that data be exchanged in real-time between computers participating in a particular game session.

However, as networking becomes more prevalent and collaboration becomes more advanced, the need arises for computer operators to use multiple collaborative applications from different vendors concurrently with the same group of computer operators to accomplish the required session tasks. An example of this is the use of one of the above-mentioned games in a session where each participant can also see and talk to each other while playing from remote locations. Each participant sees the others simultaneously through a third party multi-point video technology, such as Lucent's iCosm™ multi-point video system (iCV); and each participant can hear and talk to the others simultaneously through a multi-point audio mixer, such as Lucent's iCV™ IP Audio Bridge. This example adds many more layers of complexity to an already complex task, thus making it unusable or at least very difficult for the average computer operator. We will refer to these complex multi-application collaborative sessions from this point on as 'sessions' or individually as a 'session'.

The instant invention provides each computer operator or user a simple and straightforward way to enter into a complex multi-application real-time collaborative session with others.

The present invention is a system and method for linking multiple computer operators together in a multi-application collaborative session while simultaneously linking their shared project data and shared streaming broadband data needs.

Computer code for collaborative applications can store project data on a centralized server so many computer operators in the same or remote locations can have simultaneous access thereto. These same computer operators may also have a need to talk to and view other remote computer operators working on the project. This can be accomplished using other computer code creating streaming

data links between parties, as is well known in the art. Streaming data transports a variety of different media types from location to location, or one location to many other locations. These different types of media include but are not limited to: audio (voice, sounds, music, etc.), video (live video, movies, television broadcast, etc.), and computer generated graphics. This other computer code can be embedded in the same application or several different applications running simultaneously.

Current implementation processes are somewhat complex when linking just two computer operators into a collaborative session and allowing them to work on the same project data while they can talk to each other and see live video of each other. This process becomes significantly more complex when three or more computer operators are interacting in the same session that requires the use of two or more applications by different manufacturers. The process is an order of magnitude more complex when designed to handle millions of computer operators seeking to collaborate in many grouped sessions with specific other computer operators via video, voice, and data, using many different, collaborative computer applications. The present invention accomplishes this collaboration while making it simple for the average person to use.

The instant invention comprises a "computer interface" and a "centralized real-time catalog". The "computer interface" is front-end client software and, in the preferred embodiment, is referred to as "GOS" for global operating system. The "centralized real-time catalog" is back-end software and, in the preferred embodiment, is referred to as "RTKC" for real-time knowledge center. Also, the terms "catalog" and "index" are used interchangeably herein.

Each computer operator needs to have a computer interface that communicates via a data network with the centralized real-time catalog. The computer interface is

configured to catalog or index the applications that reside on the computer or the interface can poll the computer's operating system to determine which applications are currently on the computer that the interface has been  
5 installed on. The interface can also poll the centralized real-time catalog to find other applications it may either download or run remotely.

When a computer operator creates a new collaborative session, the interface reports to the centralized real-time  
10 index that session's network address, all required session parameters, and the server network address where that session's project data is stored. If any security data such as passwords are required, they can either be predefined or defined when the session is started.

15 Any computer operator wishing to join the session must first have the ability to run the required applications in accordance with that session's requirements. Once this is determined by the joining computer operator's interface, that interface sends its network address to and retrieves the  
20 session joining instructions from the Centralized Real-time Index. All security details and joining requests are then verified, and the computer operator's interface receives the network addresses of all related computers and computer servers for that session. The joining computer interface  
25 then launches all relating computer application code required for that session and passes the required network addresses to those applications to link to the corresponding applications on the other computers. At the same time the other computer interfaces in the session are informed that the joining  
30 computer has joined and they also provide that entrant's network address to the appropriate applications in order to recognize and interact with the new entering computer interface's applications.

#### BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention will be had upon reference to the following description in conjunction with the accompanying drawings, wherein:

Figure 1 is a block diagram of the system in accordance with the instant invention;

Figure 2 is a session operation flow diagram in accordance with the instant invention; and,

Figure 3 is a session initialization flow diagram in accordance with the instant invention.

#### 10 DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to the drawing figures, the instant invention is a system 10 and method for managing the simultaneous real-time operation of a plurality of software applications comprising a global operating system 20 and a real-time knowledge center 40. In a preferred constructed embodiment of the instant invention, the global operating system 20 is comprised of a computer readable medium having computer executable instructions resident in a plurality of client (front-end) computers. The real-time knowledge center 40 is preferably server based (back-end) and is comprised of a computer readable medium having computer executable instructions, and a centralized real-time catalog of session data.

On the back-end, the required results are achieved by creating a server process that tracks application sessions, applications used by computer operators, computer operators participating in applications sessions, etc. The real-time knowledge center 40 serves as a real-time catalog 42 or index of all systems events. Since collaborative applications written by various vendors are not developed to provide this information to the real-time knowledge center 40, those third party applications must be forced to operate in accordance with system requirements. The global operating system 20 of the instant invention provides a client-based system for

managing said third party applications, as described hereinbelow.

When the global operating system client starts, depending on the particular client system configuration, it automatically starts from 0 to n (some number) processes (application servers) on a background that hosts some applications sessions. The global operating system client then reports all process parameters to the real-time knowledge center 40 server, which updates its catalog or index 42, registering the client's network address and the applications sessions it hosts.

When a computer operator logs in to a client computer using the global operating system client interface 22, the global operating system 20 reports this event to the real-time knowledge center 40 server, which then updates its catalog or index 42 with respect to the current computer operator at that computer's network address. When that computer operator wants to use a particular collaborative application, the global operating system client requests from the real-time knowledge center 40 server a list of available sessions for a given application. When that computer operator selects a particular session to participate in, the global operating system 20 starts the applications required for that session and passes to them the parameters retrieved from the real-time knowledge center 40 so they will connect to that particular session. The information that a given computer operator has joined the session is also registered in the real-time knowledge center server catalog 42.

When the computer operator selects a collaborative session to join, in addition to other session parameters such as session name and the applications to launch, the global operating system 20 also receives from the real-time knowledge center 40 information in the form of data about the computer operators already participating in the session. This data includes all network addresses of session computer

operators' computers. When the computer operator joins the session, the global operating system 20 client software points all the relating applications toward the aforementioned network addresses.

5           For example, with three session members (players) already in a Quake III™ session and a fourth entering, the global operating system 20 on the fourth computer would launch Quake III™ and point it directly to a designated QUAKE III™ server, would launch three iCOSM™ video windows  
10           and point them individually to each of the three other session members' computers, and would point the iCV audio towards a designated iCV Audio Bridge Server. Simultaneously, the real-time knowledge center server notifies the other session members that a new computer  
15           operator has joined the session. This event causes the global operating system 20 on every session member's computer to create a new iCOSM™ video window and point it to the new session member's computer. In this example, the iCV Audio Bridge automatically permits the addition of the new session  
20           member's voice to the session so nothing has to be done to conform the voice application on the new session member's computer. Consequently, all computer operators participating in a given session can see and talk to each other while simultaneously collaborating in the game application.

25           Alternatively, the system can be configured in such a way, that if a new computer operator is of particular type (a system operator for example), that computer operator receives video/audio input from all members of the session, but none  
30           of the normal session members can see or hear that special computer operator. This embodiment of the instant invention can be used for system monitoring purposes. Also, the system can be configured to allow any computer operator or only specified computer operators, such as system administrators, to create and launch new sessions of any conforming  
35           applications.

The system and method described herein is a completely distributed system having no single data bottleneck. Application sessions are hosted on different network nodes (computers). Streaming data such as real-time voice and video streams are exchanged directly between every two client computers participating in the same session. Furthermore, the real-time knowledge center 40 exchanges only catalog or index data. The real-time knowledge center does not exchange either application specific or streaming data, so its bandwidth requirements are very low. Additionally, catalog data can be distributed on several real-time knowledge center servers, thereby further reducing the threat of any centralized data bottleneck. For example, one server can maintain catalog data related to entertainment application sessions, another related to health care applications sessions, and yet another to business applications sessions. Furthermore, catalog data can be replicated among several servers to achieve a higher level of reliability and load balancing.

The nature and variety of collaborative applications requires the global operating system 20 to force those different applications to behave uniformly. To overcome this problem, the global operating system further comprises an application wrapper 24 that is analogous to a hardware driver, but applied to a software application instead.

To illustrate the operation of an application wrapper 24, the means by which most operating system handle printing may be described. Obviously, many different printers exist. If operating systems, for example MS Windows™ or Sun Unix™, had to be updated every time a new printer was introduced, it would cause problems as described above. Instead, a printer driver comprising a separate software module that has a known specific interface for an operating system is employed to implement printing functionality for a particular printer (type). The printer hardware configuration specifics are

hidden from the operating system in a printer driver. The operating system only "sees" that the printer driver supports a standard printing interface. Consequently, the operating system does not need to be changed when a new printer is introduced. Instead, a separate software module, called a printer driver, is created either by the operating system vendor or, more often, by a printer manufacturer.

Similarly, the application wrapper 24 in the global operating system 20 has known interfaces used by the global operating system 20. The application wrapper 24 hides peculiarities of interfacing and controlling a given application from the global operating system 20. Therefore, when a new application is used within the global operating system 20, it does not require changes in the global operating system 20 client software. Instead, a novel software module, the 'application wrapper' 24 is produced. It is comprised of a first part that provides an application specific computer operator interface, a second part that controls a corresponding collaborative application, and a keyboard filter, as described further below.

The complete process of joining a collaborative session is described in the following steps as detailed in Fig. 2.

- 1) The Computer operator (user) selects a main application from applications available on a given global operating system computer.
- 2) The global operating system retrieves available sessions for the given application.
- 3) The computer operator decides on the session to join.
- 4) The global operating system 20 loads a corresponding application wrapper and requests that wrapper to start the main application required for that collaborative session and connect the application to a particular session server and session project data.
- 5) The global operating system 20 launches the corresponding required minor applications in the correct order. This accomplishes tasks like establishing data communications channels for both video and voice

streaming input from all members of a given session. 6) Once notified by the real-time knowledge center 40, all other session group members' computers also launch the corresponding applications to establish corresponding channels to transmit and receive streaming data to and from a new computer operator.

When a computer operator decides to quit a session, an opposite process occurs. The global operating system 20 client software requests the application wrapper corresponding to the main application to shutdown the main application, then it shuts down all corresponding minor applications. Next, the global operating system client informs the real-time knowledge center server that it is disconnecting from the session. The real-time knowledge center server then updates its catalog (index) and then notifies all other session member computers that the given member computer has quit the session. After receiving that notification, the global operating system 20 installed on all of the remaining session member computers shuts down all corresponding application windows related to the leaving member's computer.

In order to allow the global operating system 20 to control applications through a corresponding application wrapper 24, the global operating system client software intercepts and acts on computer operator actions before a current application receives corresponding action notifications. For example, it was determined that it would be convenient to shutdown all applications, regardless of their nature when the escape key is pressed. However, different applications behave differently when the escape key is pressed. Some exit immediately, while others prompt computer operators prior to termination, and others simply ignore this action. The global operating system 20 controls application responses to keep the real-time knowledge center catalog synchronized.

To achieve this, the global operating system client software may further comprise a keyboard driver/filter. The filter intercepts keyboard press events and allows the global operating system client software to decide what action or actions should follow, prior to the application receiving it. For example, when the escape key is pressed, the global operating system client deactivates (pauses) the current collaborative application and asks the computer operator whether he or she wants to quit a given session. If the computer operator confirms his decision, the global operating system 20 executes all steps described above to disconnect the global operating system computer from the session.

The instant invention employs a server process that maintains a real-time catalog 42 (index) of a collaborative system and client side computer readable medium having computer executable instructions that communicates with server side computer readable medium, providing the computer operator with an interface that manages complex multi-application collaborative sessions and third party applications through application wrappers.

Referring to Figs. 1 and 2, the global operating system 20 client software is installed locally on a plurality of client computers, providing a system and method to enter into a complex multi-application real-time collaborative session with other users. The instant invention provides a "public friendly" interface to the user and launches and manages all applications required for a particular session. The same primary application and the same set of supporting applications are launched by the global operating system 20 for each user in a session.

The instant invention can be implemented in conjunction with a plurality of primary applications, for example, Auto-Cad™, Quake™, Excel™, etc. Furthermore, the instant invention can be implemented by client computer systems

running a plurality of supporting applications, for example, video, voice, data, MS Word™, Excel™, etc.

The real-time knowledge center 40 server software is resident on a server platform, or a plurality thereof, and maintains a real-time catalog or index of events in the whole system, maintains session member (user) listings and client network addresses, maintains project data location addresses, maintains primary application support requirements, and maintains all application location addresses.

The real-time knowledge center 40 supplies all connection and configuration data required for a particular collaborative session to the client's global operating system 20, for example network addresses, session listings, configuration data, etc. The real-time knowledge center 40 and global operating system 20 continue to exchange session update data throughout a session to support users joining and leaving a session. Since this is change and configuration data only, bandwidth requirements are very low.

A plurality of medium to high bandwidth real-time primary application data channels are provided to exchange real-time data directly between applications. Furthermore, a plurality of high bandwidth streaming application data channels exchange are provided to exchange real-time video images and real-time audio directly between applications.

In one embodiment of the instant invention, a plurality of medium to high bandwidth real-time supporting application data channels are provided to exchange real-time data directly between applications. Any number of client computers may connect to a particular application session.

Referring to Fig. 2, a session operational flow diagram is provided with the operational steps labeled one through eight. The global operating system starts up on local users systems and the global operating system 20 reads a configuration file from local disk. The global operating

system 20 then loads and launches a plurality of application wrappers 24 for configured applications.

The real-time knowledge center 40 starts up on a server, reads a configuration file from a local disk, and initializes the dynamic catalog or index.

A user may then enter log-on data to the global operating system 20. The global operating system 20 then sends client address to the real-time knowledge center 40, which adds the client address to a list of active users in the dynamic catalog 42. The real-time knowledge center 40 then sends a log-on accepted message to the global operating system 20.

In the primary application selection sequence the global operating system 20 presents a primary application list to the user, who selects an application. The global operating system 20 then sends a primary application request to the real-time knowledge center 40.

In the session selection sequence the real-time knowledge center 40 sends an available session list to the global operating system 20. The global operating system 20 presents the available session list to the user. When the user selects a session, the global operating system 20 sends a session join request to the real-time knowledge center 40, which adds the client to the session in the dynamic catalog 42 and the real-time knowledge center 40 sends the client address to other users in session.

In the session launch sequence, the real-time knowledge center 40 fetches the session name from the catalog, fetches supporting application requirements from the catalog, fetches other user addresses from the catalog, and then sends session requirements to the global operating system 20. The global operating system 20 then launches all required applications.

In the session maintenance sequence, the real-time knowledge center 40 sends a new user addresses to the global operating system 20 as users join the session. The real-time

knowledge center 40 then notifies the global operating system 20 when users leave the session.

In the user log-off sequence, a user requests to log-off, then the global operating system 20 notifies the real-time knowledge center 40 that client has left session. The real-time knowledge center 40 then removes client from the session list in the catalog 42 and notifies other users in the session that the client has left.

Figure 3 depicts an administrator setting up a session. However, a user or users may also be permitted to set up a session. All events prior to the log-on sequence are described in the above discussion of Figure 2. In the administrator log-on sequence, an administrator enters log-on data to the global operating system 20. The global operating system 20 sends client address to the real-time knowledge center 40, which then adds the client address to active users in the dynamic catalog 42. The real-time knowledge center 40 enables administration privileges for the client and sends a log-on accepted message to the global operating system 20.

In the primary application selection sequence, the global operating system 20 presents a primary application list to a user. The administrator selects a new session startup, identifies the location of the primary application for the new session for the global operating system 20, locates the project data for the new session for the global operating system 20, and names the new session. The global operating system then sends session start request and session data to the real-time knowledge center 40. The real-time knowledge center 40 then starts up a new session in the dynamic catalog 42, and reports session requirements to the global operating system 20. The global operating system 20 then launches main and supporting applications.

In addition to normal termination of a session by the various users, a session can also be turned off in other ways. The administrator may shut the session down or set a

predetermined time to shut down the session. For example, the session could be set to end on a selected date at 5 p.m., or, the session could terminate based on user inactivity over a period of time. Other triggering events could be used to terminate a session.

The following description is an example of application wrapper 24 implementation. In this instance, application wrappers are implemented as ActiveX automation objects in a form of dynamically linked libraries (DLL). All code samples are in C++ with a use of ATL (Active Template Library), but any other programming language capable of producing ActiveX automation objects can be used.

Every application wrapper in this example supports at least IAppCore interface. This interface has methods implementing core (basic) tasks, such as starting an application session, starting an application in a single user mode (stand alone), starting an application in multi-user mode (connecting to a given session), exiting application, and so forth.

The global operating system 20 can query an application wrapper 24 to check if it supports additional interfaces using standard OLE (ActiveX) method - QueryInterface().

The following is a definition of IAppCore interface in IDL (Interface Definition Language):

```
interface IAppCore : IDispatch
{
    [id(1), helpstring("method SetPropertyApp")]
    HRESULT SetPropertyApp([in] long propertyID, [in] BSTR
        propertyValue);

    [id(2), helpstring("method GetPropertyApp")]
    HRESULT GetPropertyApp([in] long propertyID, [out] BSTR
        * propertyValue);

    [id(3), helpstring("method StartGDS")]
    HRESULT StartGDS();

    [id(4), helpstring("method StartMultipleApp")]
    HRESULT StartMultipleApp([in] BSTR networkAddress);
}
```

```

[id(5), helpstring("method StartSingleApp")]
    HRESULT StartSingleApp();

[id(6), helpstring("method ActivateApp")]
    HRESULT ActivateApp();
5 [id(7), helpstring("method DeactivateApp")]
    HRESULT DeactivateApp();

[id(8), helpstring("method CloseApp")]
10 HRESULT CloseApp();

[id(9), helpstring("method SetGUIPtr")]
    HRESULT SetGUIPtr(IUnknown * pGui);

15 [id(10), helpstring("method StopGDS")]
    HRESULT StopGDS();

[id(11), helpstring("method GetKeyBlockMap")]
    HRESULT GetKeyBlockMap([in] VARIANT* varKeys);
20 };

```

The example below is an exemplary application wrapper implementation for a Quake III™ program.

The global operating system 20 passes application parameters it gets from a configuration file or/and from user interface module to an application wrapper 24 without any understanding of the nature of those parameters. To the global operating system 20, this is simply a (property id, property value) pair. The interpretation of these parameters are accomplished by the application wrapper 24 and, ultimately, by the application itself.

```

STDMETHODIMP CAppCore::SetPropertyApp(long propID, BSTR
propValue)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())
35 CString strValue = propValue;

    if (CAgoraAppBase::SetPropertyApp(propID, strValue))
    {
40         return S_OK;
    }

    switch(propID)
    {
45     case PROP_QUAKE_APP_PATH_ID:
        m_AppPath = strValue;
        break;

```

```

case PROP_QUAKE_APP_NAME_ID:
    m_AppName = strValue;
    break;
case PROP_QUAKE_SAVE_LEVELS_PATH_ID:
5     m_SaveLevelsPath = strValue;
    break;
case PROP_QUAKE_START_PARAM_ID:
    m_StartParam = strValue;
    break;
10    case PROP_QUAKE_SINGLE_LEVEL_PARAM_ID:
        m_SingleLevelParam = strValue;
        break;
case PROP_QUAKE_SKIN_PRIMARY_PARAM_ID:
    m_SkinParamPrimary = strValue;
15    break;
case PROP_QUAKE_WAIT_FIND_WND_ID:
    m_SecWaitFindWnd = atoi((const char
*) (LPCTSTR)strValue);
    break;
20    case PROP_QUAKE_CONFIG_FILE_PATH:
        m_ConfigFilePath = strValue;
        break;
case PROP_QUAKE_GDS_START_PARAM_ID:
    m_GDSStartParam = strValue;
25    break;
default:
    return(S_FALSE);
}

30    return S_OK;
}

```

The method below starts a background process (Quake III™ Server) which is responsible for implementing a Quake III™ game session. Note that some session parameters are based on information passed by the SetPropertyApp() method.

```

STDMETHODIMP CAppCore::StartGDS()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    // Set current work directory (if exist) for given app
    if (!m_AppPath.IsEmpty())
        SetCurrentDirectory(m_AppPath);

    ////////////////////////////////////////
    // Start Quake3
    STARTUPINFO startupInfo;
    memset(&startupInfo, 0, sizeof(STARTUPINFO));
    startupInfo.dwFlags |= STARTF_USESHOWWINDOW;
50    startupInfo.wShowWindow = SW_HIDE;
}

```

```

startupInfo.cb = sizeof(STARTUPINFO);
BOOL res = FALSE;

```

```

5   try
    {
        res = CreateProcess(m_AppName,      (LPTSTR) (LPCTSTR) (" "
+
                                     m_GDSStartParam + " " + m_StartParam
                                     + " " + GDS_CONFIG_FILENAME), NULL,
10  NULL, FALSE, 0, NULL, NULL,
                                     &startupInfo, &m_ProcessInfo);
    }
    catch(...)
    {
15      return S_FALSE;
    }

    if (!res)
        return S_FALSE;
20  return S_OK;
}

```

25 The exemplary method shown below starts the Quake III™ application and connects it to a particular game session based on a given network address. Note that in this particular situation, the network address is just an IP (TCP/IP) number representing the computer where a corresponding Quake III™ server resides. In alternative
30 embodiments, the network address can be an IP number and an IP port number, or any other data needed to establish connection with a particular session.

```

STDMETHODIMP CAppCore::StartMultipleApp(BSTR networkAddress)
35 {
    AFX_MANAGE_STATE(AfxGetStaticModuleState())

    // Set current work directory (if exist) for given app
    if (!m_AppPath.IsEmpty())
40      SetCurrentDirectory(m_AppPath);

    // Delete all from Save directory
    if (!m_SaveLevelsPath.IsEmpty())
        EmptyDirectory(m_SaveLevelsPath);
45

    // Create cfg file
    CString fName = m_ConfigFilePath + "\\\" +
    MULTIPLE_CONFIG_FILENAME;

```

```

CString cfgStr;

try
{
    5   CStdioFile cfgFile(fName, CFile::modeCreate |
CFile::modeWrite);

        // Set connection string
        CString strStationIP = networkAddress;
    10   if (!strStationIP.IsEmpty())
        {
            cfgStr.Format((LPCTSTR)"connect %s\n",
strStationIP);
            cfgFile.WriteString(cfgStr);
    15   }

        cfgFile.Close();
    }
    catch(...)
    {
    20   return S_FALSE;
    }

    25  ////////////////////////////////////////////
        // Start Quake3
        STARTUPINFO startupInfo;
        memset(&startupInfo, 0, sizeof(STARTUPINFO));
        startupInfo.dwFlags |= STARTF_USESHOWWINDOW;
    30   startupInfo.wShowWindow = SW_HIDE;
        startupInfo.cb = sizeof(STARTUPINFO);
        BOOL res = FALSE;

        CString csStartParameter = "";
    35   // Set user name
        if (!m_UserName.IsEmpty())
        {
            csStartParameter.Format((LPCTSTR)" +set name %c%s%c",
    40   DOUBLE_QUOTE_CHR, m_UserName, DOUBLE_QUOTE_CHR);
        }

        // Set skin
        if (!m_SkinParamPrimary.IsEmpty())
    45   {
            csStartParameter += CString(" +set model ") +
                CString(DOUBLE_QUOTE_CHR) +
                m_SkinParamPrimary +
                CString(DOUBLE_QUOTE_CHR);
    50   }

        csStartParameter += " " + m_StartParam + " " +

```

MULTIPLE\_CONFIG\_FILENAME;

```
try
{
5   res = CreateProcess(m_AppName,
(LPTSTR)(LPCTSTR)(csStartParameter),
                                NULL, NULL, FALSE, 0, NULL, NULL,
                                &startupInfo, &m_ProcessInfo);
10  }
    catch(...)
    {
        return S_FALSE;
    }
15  if (!res)
        return S_FALSE;

    CWnd* pMainWnd = FindAppMainWnd(m_ClassName,
20  m_SecWaitFindWnd);

    if (pMainWnd == NULL)
        return S_FALSE;

25  // Change window style
    long style = GetWindowLong(pMainWnd->GetSafeHwnd(),
    GWL_STYLE);
    style ^= WS_CAPTION;
    SetWindowLong(pMainWnd->GetSafeHwnd(), GWL_STYLE, style);
30  // Change window position
    pMainWnd->SetWindowPos(&CWnd::wndTopMost, 160, 120, 640,
480, 0);
    pMainWnd->ShowWindow(SW_SHOW | SW_RESTORE);
35  // If we did not pass a callback interface (did not call
    AtlAdvise())
    // it does not make sense even start a thread
    IUnknown **ppUnk = m_vec.begin();
40  if (*ppUnk != NULL)
    {
        _IAppCoreEvents * pCallback = (_IAppCoreEvents *)
        *ppUnk;
45  StartCheckAppThread(m_ClassName, pCallback);
    }

    return S_OK;
}
```

50 Each application wrapper also accepts a callback interface provided by the global operating system, which

serves as a notification mechanism to pass information back from the application wrapper to the global operating system. An exemplary callback interface supporting one method is shown below:

```
5 interface _IAppCoreEvents : IDispatch
{
    [id(1), helpstring("method OnAppClosed()")] HRESULT
    OnAppClosed();
};
```

10 This event serves as a notification in the event the application stops running without a global operating system request, for example, in an application crash.

The instant invention may also be employed as a method of using remote live experts on the the global operating system/real-time knowledge center system. These experts, alone or in teams, interact with the computer operators over the computer network via voice, video and data to help the computer operators with the many complexities of the individual applications residing in the global operating system or any other needs that may arise. On the global operating system/real-time knowledge center system, these experts may act as, but are not limited to: a routing operator to other experts, a guide, a tournament host, a friend, a professional advisor (for example a doctor, a lawyer, a stock broker, a teacher), a teammate, an opponent, a referee, a help desk, a complaint desk, a customer service representative, a retail sales representative, and a new product marketer.

Even though several of these types of experts exist today on the world-wide web, they have never before interacted over any network in a highly collaborative multi-participant, voice, video and data format. Consumers will benefit by being able to interact over the global operating system/real-time knowledge center system in an audiovisual format with factory direct experts when making purchasing decisions or solving problems with purchased merchandise.

The foregoing detailed description is given primarily  
for clearness of understanding and no unnecessary limitations  
are to be understood therefrom for modifications can be made  
by those skilled in the art upon reading this disclosure and  
5 may be made without departing from the spirit of the  
invention and scope of the appended claims.